Contents

Extensibility

2 minute read

WebAssembly is a sandboxing technology which can be used to extend the Istio proxy (Envoy). The Proxy-Wasm sandbox API replaces Mixer as the primary extension mechanism in Istio.

WebAssembly sandbox goals:

- **Efficiency** - An extension adds low latency, CPU,

and memory overhead.

- **Function** - An extension can enforce policy, collect telemetry, and perform payload mutations.
- **Isolation** - A programming error or crash in one plugin doesn’t affect other plugins.
- **Configuration** - The plugins are configured using an API that is consistent with other Istio APIs. An extension can be configured dynamically.
- **Operator** - An extension can be canaried and deployed as log-only, fail-open or fail-close.
- **Extension developer** - The plugin can be written in several programming languages.

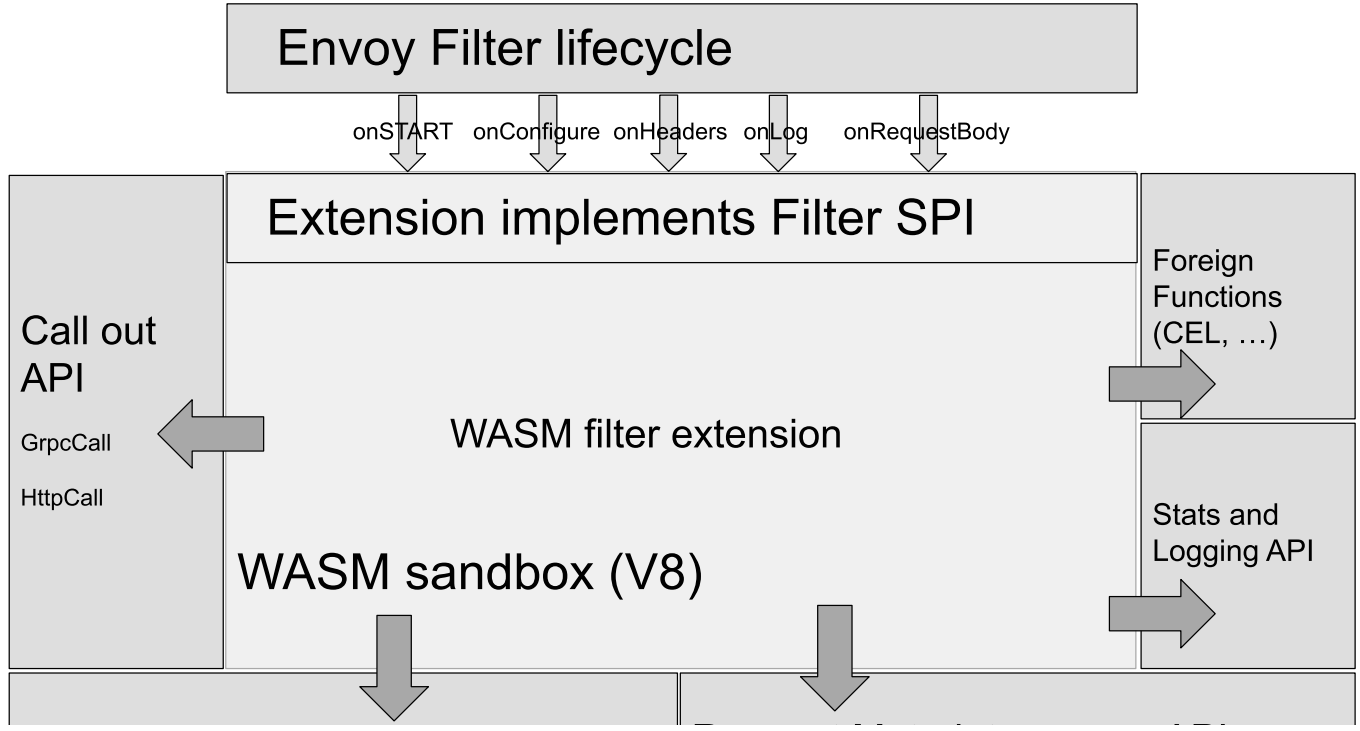
This video talk is an introduction about architecture of WebAssembly integration.

High-level architecture

Istio extensions (Proxy-Wasm plugins) have several components:

- **Filter Service Provider Interface (SPI)** for building Proxy-Wasm plugins for filters.
- **Sandbox** V8 Wasm Runtime embedded in Envoy.

- **Host APIs** for headers, trailers and metadata.
- **Call out APIs** for gRPC and HTTP calls.
- **Stats and Logging APIs** for metrics and monitoring.



Header and Trailer access API getRequestHeaders, setRequestHeaders…	Request Metadata access API getRequestMetadata, setRequestMetadata…

Example

An example C++ Proxy-Wasm plugin for a filter can be found here. You can follow this guide to implement a Wasm extension with C++.

Ecosystem

- Istio Ecosystem Wasm Extensions
- Proxy-Wasm ABI specification
- Proxy-Wasm C++ SDK
- Proxy-Wasm Rust SDK
- Proxy-Wasm AssemblyScript SDK
- WebAssembly Hub
- WebAssembly Extensions For Network Proxies (video)